

# توقع أسعار العملة التركية مقابل الدولار

الإنحدار الخطي

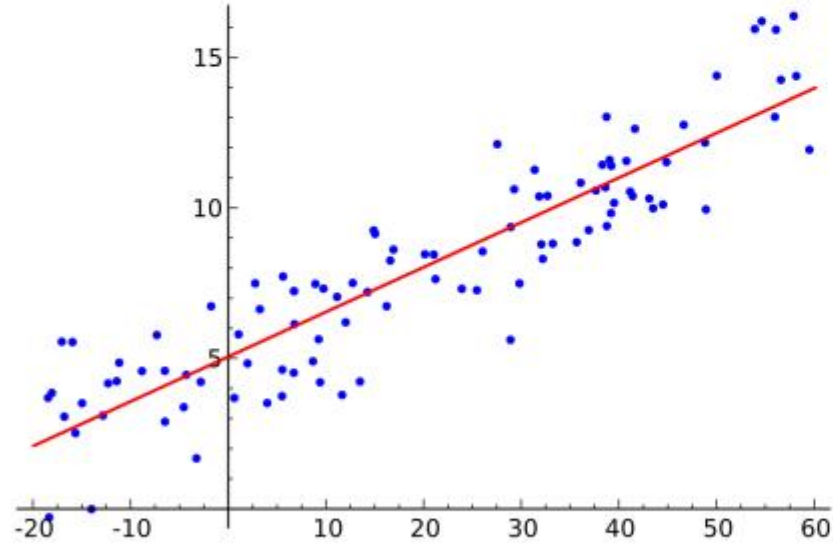
# تعريف الإنحدار الخطي

• الإنحدار الخطي هو خوارزمية إحصائية تستخدم لنمذجة العلاقة بين متغيرين. تفترض هذه الخوارزمية أن هناك علاقة خطية بين المتغير التابع (المتغير الذي يتم توقعه أو شرحه) ومتغير واحد أو أكثر من المتغيرات المستقلة (تلك المستخدمة لإجراء التنبؤ). الهدف من الإنحدار الخطي هو العثور على الخط الأنسب الذي يصف العلاقة بين المتغيرات. حيث يتم تحديد أفضل خط مناسب بتقليل مجموع الفروق التربيعية بين القيم الفعلية والقيم المتوقعة.

• الإنحدار الخطي نوعان: الإنحدار الخطي البسيط والإنحدار الخطي المتعدد. يتميز الإنحدار الخطي البسيط بمتغير مستقل واحد. بينما يتميز الإنحدار الخطي المتعدد بالعديد من المتغيرات المستقلة (أكثر من واحد) أثناء البحث عن أفضل خط مناسب.

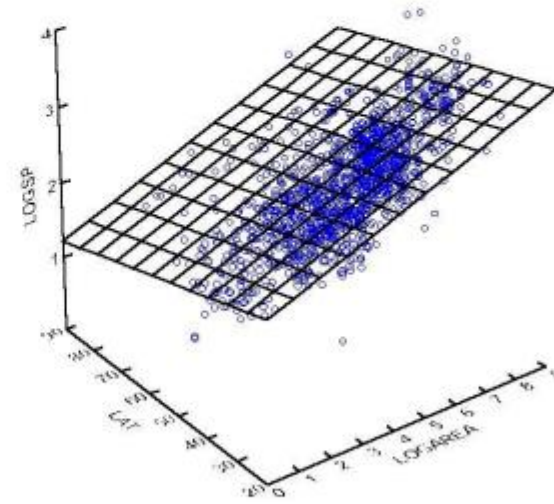
# أنواع الإنحدار الخطي

الإنحدار الخطي البسيط



$$y = bX + c$$

الإنحدار الخطي المتعدد

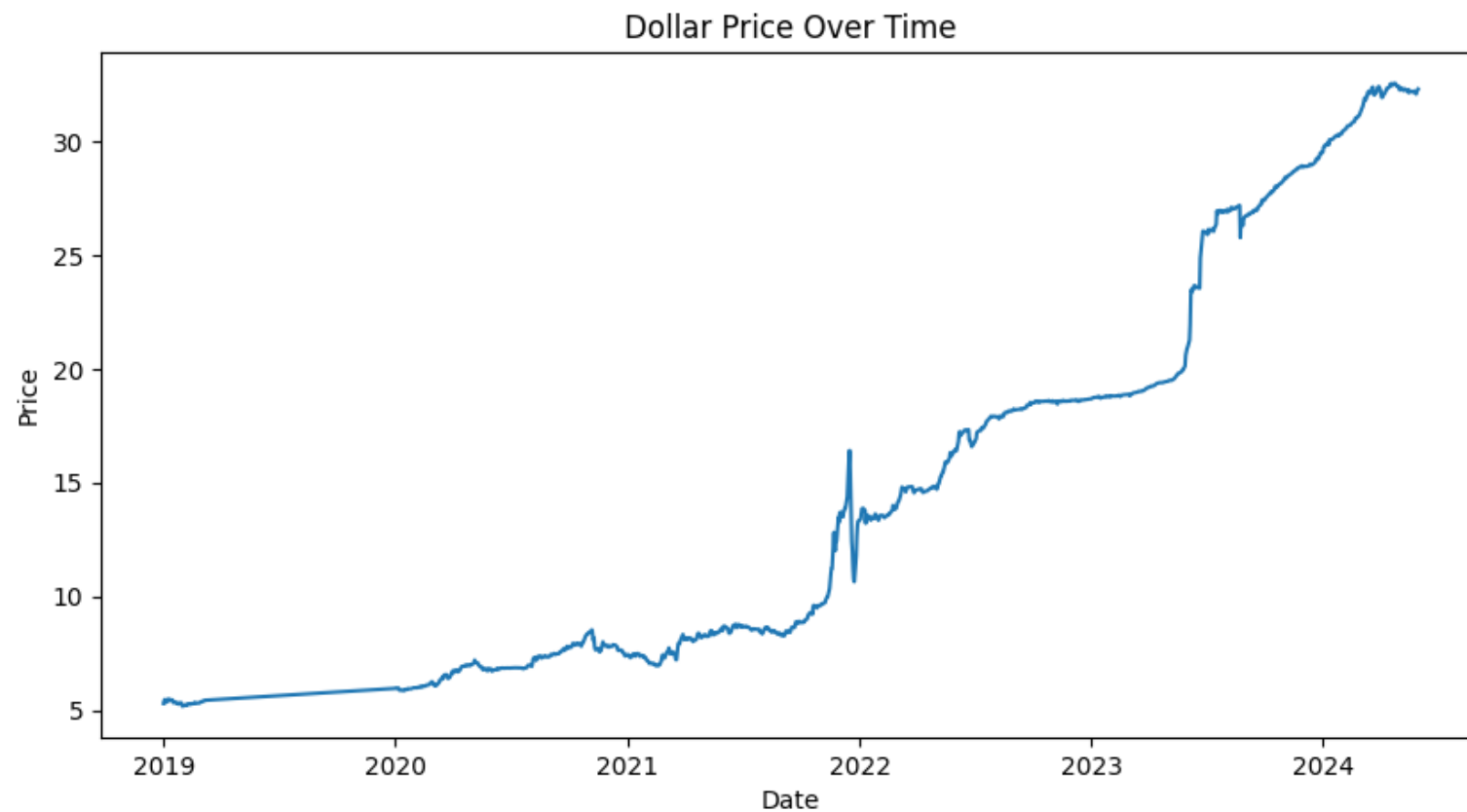


$$y = b_1x_1 + b_2x_2 + \dots + b_nx_n + c$$

- الخطوة الأولى هي الحصول على البيانات التي سيتم تحليلها. في هذه الحالة، لدينا ملف CSV يحتوي على تاريخ وسعر الليرة التركية مقابل الدولار. مصدر الملف

Date	Price	Open	High	Low	Vol	Change %
05/30/2024	32.3217	32.2648	32.3323	32.1917		0.20%
05/29/2024	32.258	32.2470	32.3050	32.1575		0.03%
.....	.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....	.....
01/01/2019	5.2933	5.2896	5.2933	5.2540		0.09%

# مخطط سعر العملة التركية مقابل الدولار منذ عام ٢٠١٩



برنامج تحليل وتنبؤ بسعر الليرة التركية مقابل الدولار الأمريكي باستخدام مكتبات بايثون المختلفة مثل:

**Pandas** لمعالجة البيانات.

**Numpy** للتعامل مع المصفوفات والأعداد.

**Matplotlib** لرسم المخططات البيانية.

**train\_test\_split** لتقسيم البيانات إلى مجموعتي تدريب واختبار.

**ARIMA** نموذج لتحليل السلاسل الزمنية (لا يستخدم فعليًا في الكود).

**LinearRegression** لبناء نموذج الانحدار الخطي.

**mean\_squared\_error** لحساب خطأ التربيع المتوسط لتقييم النموذج.

# إنشاء كلاس UsfToTRY للتنبؤ بالسعر

• قراءة البيانات:

```
class UsdToTRY:  
    def __init__(self) :  
        # self.data = pd.read_csv('USD_TRY Historical Data.csv')  
        self.data = pd.read_csv('USD_TRY Historical 6-years.csv')
```

- فحص البيانات: لعرض الصفوف الأولى من البيانات، الإحصائيات الأساسية، والتحقق من القيم المفقودة.

```
def checkData(self):  
    # Display the first few rows  
  
    print(f'head {"-"*100}\n{self.data.head()}')  
  
    # Basic statistics  
    print(f'describe {"-"*100}\n{self.data.describe()}')  
  
    # Check for missing values  
    print(f'is Null {"-"*100}\n{self.data.isnull().sum()}')
```



معالجة البيانات:

تحويل عمود Date إلى صيغة تاريخية.

إزالة القيم المكررة.

ترتيب البيانات حسب التاريخ.

استخراج اليوم، الشهر، والسنة من عمود التاريخ.

إزالة عمود Vol. الفارغ.

إزالة الصفوف التي تحتوي على قيم مفقودة.

تعيين المتغيرات المستقلة (X) والمتغير التابع (Y)

```
def dataPreprocessing(self):  
    self.data['Date'] = pd.to_datetime(self.data['Date'])  
  
    self.data = self.data.drop_duplicates(subset=['Date'])  
    self.data =  
self.data.sort_values(by='Date', ascending=False)  
    self.data['day']=self.data['Date'].dt.day  
    self.data['month']=self.data['Date'].dt.month  
    self.data['year']=self.data['Date'].dt.year  
    #drop Empty Column  
    self.data=self.data.drop('Vol.', axis=1)  
    self.data.dropna(inplace=True)  
    self.X=self.data[['day', 'month', 'year']]  
    self.Y=self.data['Price']
```

# بناء وتدريب النموذج + تقييم النموذج

- `def buildAndTrainTheModel(self):`
- `# Initialize the model object`
- `self.model = LinearRegression()`
- `# Training the model`
- `self.model.fit(self.X, self.Y)`

```
• def evaluateTheModel(self):  
•     # Make predictions  
•     # print(self.X_test)  
•     y_pred = self.model.predict(self.data[['day', 'month', 'year']])  
•     # Calculate the mean squared error  
•     mse = mean_squared_error(self.Y, y_pred)  
•     print(f'Mean Squared Error: {mse}')
```



```
•     # Plot actual vs predicted prices  
•     plt.figure(figsize=(10,5))  
•     plt.plot(y_pred, label='Predicted Price')  
•     plt.plot(self.Y, label='Actual Price')  
•     plt.title('Actual vs Predicted Dollar Price')  
•     plt.xlabel('Time')  
•     plt.ylabel('Price')  
•     plt.legend()  
•     plt.show()
```